Embedding prior phonetic knowledge in a text-to-speech system using data driven methods.

Technical Report

Andrew D. Cohen,

Department of Cybernetics,

University of Reading.

October 17, 1996

Abstract

Research described here explores ways of incorporating prior knowledge about a phonetic problem (here text-to-speech synthesis) into data-driven systems (here feed-forward neural networks), which operate in a 'black box' manner opaque to interpretation, and which start from a tabula rasa and the data set alone. As many authors in the neural network field have noted, it is better to focus on issues such as representations and prior knowledge rather than the learning dynamics, since there are no general purpose learning algorithms which are good for an arbitrary prior (Wolpert and Macready, 1995) (see §3 for discussion of the term 'prior'). Hence, there is a need to introduce domain specific knowledge (here phonetic/phonological knowledge) in modules that can influence the development of other modules, or modify their output. Phonetic knowledge has been introduced using active means: knowledge transfer from one network to another, and static means: learning a representation for target patterns and using an error-correcting network. Two learning procedures (using delayed feedback from the output units and training an encoder network with noise) have been applied to correct a main network that derives phonemes (and eventually speech) from text. Both these procedures include a period of pre-training on output patterns only, before the mapping from inputs to outputs is learned. The output representations are based on a phonetic notation D_{phon}, which itself is derived by a self-organising process. In successive stages text is mapped onto phonemes, then diphones, using two phonetic spaces D_{phon} and D_{dip} that have been learned by Self-Organizing Maps (SOMs; Kohonen, 1989;1990). Results show that the pre-training and output representations used allow a standard 3-layer backpropagation network to improve its overall performance, and to reduce consonant for vowel and vowel for consonant substitution errors. Results are also presented for a set of 10 multiple networks trained on different partitions of a 20,000 word data set, whose outputs are combined by simple averaging and weighted averaging. The latter results show that the learned output representations give a useful performance improvement with averaging, though not with majority voting, suggesting that error correction for this task is better applied at a co-ordinate level (in the D_{phon} space) than at the phoneme (class) level. Various error correcting procedures are investigated in the context of multiple networks

1. Introduction

The text-to-phoneme problem has been tackled using a variety of schemes from the neural network and machine learning communities (Sejnowski and Rosenberg, 1987 (S&R); van den Bosch, 1993; Dietterich et al, 1995). Generally speaking the data-driven approaches have not attempted to make use of linguistic knowledge, by which is usually meant morphological and phonotactic structure specific to a given language. However, there are other, non-symbolic ways of introducing phonetic structure using data-driven methods. For example, by using a continuous, ordered phonetic space to generate target values for phonemes or diphones a phonetic distance measure can be introduced (Cohen, 1995), here referred to as D_{phon} and D_{dip} respectively. Clearly human beings who learn to convert text to speech have prior knowledge of the sound system of their language (e.g. not all phonetic strings are possible), as they are speakers of a particular language before they learn to read it. The architecture presented here incorporates built in phonetic structure through a Self-Organizing Map (SOM), which also introduces a transparent phonetic level of operation, and an Auto-Associative (AA) network which develops attractor basins to improve robustness on large data sets. Results are given for training using the SOM notation with and without the AA circuit, and with a randomized set of target coordinates as a control. Results with multiple networks trained on different data show that use of the D_{phon} space is beneficial when the networks are combined with averaging, but not with majority voting. However, this work does not fall within the machine learning tradition because it attempts to incorporate knowledge from outside the data set.

The motivation for the work described here is to build a phonological knowledge base using the D_{phon} notation, which is learned prior to the main grapheme-phoneme task being learnt, and which is then built on by additional modules. The latter has been carried out with corrector networks (§5.1), a rudimentary form of knowledge transfer (§5.2) and through an ambiguity measure derived from multiple networks (§5.2). This system has been built to test the viability of the architecture and training methodology, and is not yet competitive with rule-based systems such as DECtalk or with diphone systems containing more than 1200 diphones. Some introductory material on bias-variance, neural networks and combining multiple networks is included, but no derivations are given for the formulas used. The latter can be found in the sources cited.

2. Mapping Graphemes onto Phonemes

Learning a grapheme-phoneme mapping has typically been viewed as a multi-class classification problem, probably because each phoneme (class of sounds) has been viewed as a discrete entity. The Boolean nature of most phonetic features (and the use of phonemes themselves) leads very naturally to the task being viewed as learning a very complex discrete mapping between sets of binary strings. S&R introduced a distributed output encoding consisting of a string of binary phonetic features. The actual output was the closest of the 324 valid binary strings, determined using the angle between the two vectors. In this way the nature of the discrete mapping onto a limited set of classes is maintained, although the output space would contain 2^{26} strings if all the unpermitted combinations of phonetic features were allowed. One well-understood way to generate this kind of distributed encoding is through error-correcting codes, which have long been used in the machine-learning literature. The use of these codes requires considerable insight into the problem at hand - for example the process of achieving a good separation in Hamming space of each codeword for a specific problem has by no means been automated, and hand-correction may be necessary. As noted in Dietterich and Bakiri (1991) performance may improve as the length of the code increases, but the transparency of the results is correspondingly diminished. A detailed comparison between ID3 and Backprop has been made in Dietterich et al, (1990), and with error-correcting codes in Dietterich and Bakiri (1995). However, these codes are best suited to multi-class classification problems, and cannot be used when generalization (in the sense of function approximation) is sought, requiring that outputs not in the training set be acceptable as correct answers. The latter requires interpolation, something tree-based classifiers (such as ID3) are not capable of. In NETtalk the output was the nearest correct (i.e. phonetically permitted) pattern, where the distance measure was either Euclidean or based on the angle between the vectors (both gave similar results). If a suitable metric space can be learned from features known to be significant in the domain, then many of these difficulties can be avoided, and the metric may be adapted to the task. In this study the Kohonen map has been used to introduce a phonetic distance measure based on prior knowledge of a minimal set of features which can distinguish the phonemes.

Additionally, if the two domains (text and phonemes) are viewed as essentially symbolic, a question arises as to whether a sub-symbolic, data-driven procedure is appropriate for this task. An alternative is to view the phonetic domain (the output) as a continuous phoneme space, in which points lying between phonemes represent intermediate sounds (though not impossible combinations of binary phonetic features). In this way the task, and the view of what counts as successful generalization, can be recast in terms of function approximation, where interpolation may generate entirely new allowable outputs not seen in the original training set. Prior knowledge of likely or 'good' outputs is incorporated in SOMtalk through techniques such as attractor basins and use of a phonetic metric (D_{phon}) in the output pattern space which build structure into the network, rather than pruning or massaging of the data set.

It is not intended to out-perform state of the art rule-based systems for converting text to speech, but rather to demonstrate the viability of a method (building on existing knowledge -here a trained network) in a phonological task. It is hypothesized that success in this task, as well as in the more general problem of modelling variability, is a matter of using good representations rather than complex computations. The methodology deployed here does not rely on knowing a good representation for a task in advance, but on learning in one or more modules which then transfer their capabilities to other modules in the system, or which improve the performance of the latter in some way. Future work will extend the method (using pre-structuring and pre-training) to work directly with the spectral parameters of the diphones themselves in the form of log area ratios of the acoustic tube representation.

2.1 Data set and pronunciation dictionary

The data set is based on Sejnowski and Rosenberg's (1987) set of nearly 20,000 words, converted to a Southern British English pronunciation using the Oxford CUVOALD, a machine-readable dictionary. The S&R database has the useful property of being aligned so that the number of symbols in the input and target are equal, with silent letters mapped onto the null phoneme. This alignment of symbols has been taken over, although the CUVOALD notation contains many two-letter symbols, and so has been converted into a notation which uses only one letter symbols (See appendix 1). The system may then be used to drive a diphone concatenation system for British English. In some ways the text-to-phonetics task is more difficult for British English due to the greater number of irregular spellings, and the fact that the schwa (neutral vowel) seems to be less common.

The adapted CUVOALD notation is then converted to one derived from a phoneme set created by the author, based on a set of 10 pseudo-articulatory features which gave 35 phoneme symbols. These features themselves are not used, except indirectly through the phonotopic SOM derived from them. Actual target values are generated from the co-ordinates of a 12x8 SOM. The stress information is discarded as the diphones were segmented from nonsense words and as far as possible have a neutral stress. Simplifying the problem in this way also allows successful convergence with many fewer hidden units than used in previous studies (Dietterich et al 1990, S&R (1987)) -convergence can be achieved with as few as 40 hidden units on a training set of 4000 words. The training sets are taken from the S&R database, not from continuous text, so the stress markings would in any case be subject to modification by prosodic factors. Prosody, specifically the specification of an intonation contour, is considered to be a separate matter best handled with symbolic representations.

2.2 Encoding of inputs and outputs

A moving window of 10 input letters is used, giving 220 inputs to the network using 22 bits to encode a letter. In NETtalk 26 bits per letter were used in a 1-in-n encoding scheme, which is here modified by using 21 bits + 1 bit for the five vowels a, e, i, o, u, giving 22 bits per letter, and hence 220 inputs for a 10 letter input vector. For each vowel the vowel bit is set, plus one other; for each consonant 1 bit is set in a 1-in-n scheme. A space is encoded as all zeros. In the 'letter space' thus created vowels and consonants are separately clustered and within these two clusters each point is at an equal hamming distance. A "pre-training" training set of 4000 words gave 26560 patterns, the full 19,988 word training set gave 166160 patterns, though obviously only half as many would be obtained with a window shifted by two letters/phonemic symbols per step.

The choice of output encoding is of particular importance. Output is the two phonemes for the central two letters of the window. The output generates two sets of co-ordinates on a 12 x 8 phoneme SOM, hence 20 units for each (x, y) co-ordinate pair in a 1-in-n encoding scheme gave 40 units in the output layer (see appendix 2 for the full list of co-ordinates). Various schemes have been tried for encoding coordinates, including an "equilateral" scheme to eliminate the large number of zeros that typically occur in 1-in-n encoding schemes, but little or no improvement in speed of learning or test set performance was obtained.

The pattern set is generated by taking each word and its phonemic transcription, and advancing them through the 8 character input window and 1 character at a time so, for example, the word 'government' (phonetic transcription 'gVv@nm@nt' in my notation, 'Xg gu uv vuh uhn nX Xm muh uhn nX Xt tX' in diphones) would produce 9 patterns as follows:

```
Input Window   Output Window  Co-ordinates in phoneme space
----------     -----------    ---------------------
  ->|  |<-
.... go vern      ->   gV          (6,8)  (4,1)
...g ov ernm      ->    Vv         (4,1)  (11,1)
..go ve rnme      ->    v-         (11,1)  (2,2)
.gov er nmen      ->    -uh        (2,2)  (1,3)
gove rn ment      ->    uhn        (1,3)  (7,1)
over nm ent.      ->    nm         (7,1)  (8,8)
vern me nt..      ->    muh        (8,8)  (1,3)
ernm en t...      ->    uhn        (1,3)  (7,6)
rnme nt ....      ->    nt         (7,6)  (5,4)
```

The symbol '-' represents the null phonetic character, used for letters which do not map directly to a sound, following the alignment method of S&R.

S&R pioneered the use of a distributed representation for the outputs, which are well known to have many useful properties in neural network training. Their representation used a 26-bit code based on 21 articulatory phonetic features (position of tongue, whether fricative, nasal, liquid, glide and so on) and a further 5 bits were used to encode stress information. In this way similar phonemes have similar target vectors, but there is no attempt to regulate exact distances between phonemes (e.g. the Hamming distance is meaningless in their encoding, as is the magnitude of each vector), and the overall complexity of the encoding is high. Non-phonetic features such as 'pause', 'full-stop' and 'silent' are introduced to deal with features which are essentially prosodic, and are really on a different level to the purely phonetic articulatory features. This information is left out of the problem as treated here: no attempt is made to learn prosodic features such as intonation, stress and so on, which are probably better handled with a symbolic procedure and a specialised mark-up language on the text in question.

3. Methods of encoding phonological knowledge

The term 'prior' has most often been used to refer to probability distributions (either over a set of models or parameter values of a particular model) based on the principle of indifference or a subjectively applied criterion, which are later to be modified in a Bayesian manner in the light of evidence. In either case, it is often not clear what is the precise meaning of these mathematical terms, or how they relate to the symbolic or linguistic level. The sort of reasonably sharp prior distribution required by the Bayesian method is typically not available for speech data, and we do not wish to make the assumption of a closed sample space. In the absence of a physical model of the text-phoneme conversion process, we have only the prior phonetic/phonological knowledge of the researcher. Prior knowledge in this case comes from other data sets which are also known to be representative of the task to be learned. Ideally, we wish to learn the representations themselves, but we have to start with the data in some form (there being no data independent of a particular representation). Therefore, we use the idea of error correction but with a learned representation and distance metric, on an initial phonetic representation D_{phon}.

The kind of phonological knowledge to be introduced exists at the diphone level in terms of likely combinations of phonemes to occur together, and at a word level in terms of likely sequences of diphones. Phonetic knowledge is encoded here using SOMs and a pre-trained encoder network (the AA circuit). The re-usability, or modularity, of this knowledge lies in the method by which the data is stored, rather than in the specific data used. Adopting this perspective leads to viewing many of the problems in speech synthesis as problems of deriving a workable representation by a learning process, rather than attempting to specify one a priori. Once this has been done it is natural to want to apply this encoded knowledge to similar tasks, such as the acquisition of specific vocabularies for different applications. This might be referred to as the knowledge transfer problem, discussed in §5.2 below.

The initial output representation for target vectors used in these experiments is based on a 2D encoding of phonetic features by a SOM, which satisfies all the conditions of a metric space, namely:

(a) $d(\mathbf{x}, \mathbf{y}) \geq 0$; if and only if $\mathbf{x} = \mathbf{y}$, then $d(\mathbf{x}, \mathbf{y}) = 0$

(b) $d(\mathbf{x},\mathbf{y}) = d(\mathbf{y}, \mathbf{x})$

(c) $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z})$

The Euclidean metric is used here to measure mean squared error in the phonetic space, ie for vectors with n components

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Speed of computation is not an issue, since overall calculation of distances represents only a tiny part of computation in both training and on-line modes. For creation of target patterns the co-ordinates of this Euclidean space are represented as binary vectors in a 1-of-n encoding scheme, but the same conditions hold. If a wrap-around kernel has been used in the creation of the SOM, it is advisable to use a Gray code to eliminate boundary effects from the coordinate space, though the practical effects of the wrap-around SOM have been small in the experiments run so far.

Error can then be measured meaningfully in a continuous way, as well by the percentage of incorrect symbols. Correction can be performed in the manner of S&R, as in error correcting codes, where the closest codeword to an input vector is chosen. However, once the move to a continuos phonetic output space is made it is also possible to set up other systems of error correction based around an auto-associative network (see §5.1 below), which can operate at the level of output activations as well as phonemes. Correction can be performed by attractor basin in the case of single networks. With multiple networks additional options are available, because we can measure ambiguity (network disagreement) on unlabelled data.

3.1 Interpretation of output at phoneme stage

Final output from the MLP stage, after passing through the AA circuit, consists of a binary string which is to be interpreted as the co-ordinates of two phonemes. This string is divided in two, for one segment per phoneme. The possibilities for each string are:

1. The output string is well-formed, i.e. it is successfully converted into co-ordinates,
(a) and the co-ordinates refer to a specific phoneme, or
(b) the co-ordinates refer to a location not occupied by a phoneme.

2. The output string is not well-formed, i.e. it contains too many or two few 1s and does not indicate a specific location, in which case
(a) it is converted to the nearest well-formed string, by Hamming distance.
(b) it is considered to refer to two or more locations (if there are too many 1s), then the units with the highest activation may determine the location, or an average of the two locations may be taken if the activations are of similar strength.
(c) it designates no location at all if there are too few locations active - total failure mapped to the null.

Each case can be handled with a variety of different error-correction methods, based on the particular distant metric selected (in the phoneme space or output activations space). In fact, case 2 was found to rarely occur. Case 1a accounted for the majority of the outputs and case 1b can be handled either by choosing the nearest phoneme (by Euclidean distance) co-ordinate which does label a phoneme, or, by choosing the phoneme which most strongly excites the SOM at that co-ordinate. In addition further correction is possible by passing candidate phonemes onto the next stage (diphones) to resolve. If, for example, unknown phoneme x followed by /k/ is output, and x is equidistant to /m/ and /e/ but on the diphone SOM no such transition as /m//k/ exists, then the correct output can be determined to be /e//k/. Similarly with VV transitions, which tend only to exist in continuous sentences.

There are many other methods of determining the final output given that two candidate phonemes are available for all positions except the first. Diphones not in the synthesiser set can be discarded, reducing the possible choices somewhat.

4. Bias, Variance and Model Selection

Assuming the output representations and distance metric have been determined it is then possible to define the bias and variance of a model which may fit the data. In statistical terms the precise meaning (it also has a generic meaning somewhat similar to the machine learning usage) of bias is the expected value of $(T-\theta)$, where T is an estimator of the parameter θ. The expected value of a function of a continuous random variable X over the range [a, b] is defined as

$$E[f(X)] = \int_b^a f(x)P(x)dx \qquad (1)$$

If the probabilities are all equal, then we have E(X) equal to the mean. The variance of the distribution is given by $E[(X-E(X))^2]$. There is usually held to be a trade-off between the bias and variance of a model (Geman et al, 1992). The latter have decomposed mean-squared error of an estimator (here a feed-forward network) on a data set D into bias and variance terms as follows:

$$E_D[f(I;D) - E[T|I])^2] = (E_D[f(I;D)] - E[T|I])^2 + E_D[(f(I;D) - E_D[f(I;D)])^2] \qquad (2)$$

where the input I and target T are treated here as scalars for simplicity. E_D refers to the expectation with respect to the data set D. The regression E[T|I] is the best fit in the mean squared error sense, so the LHS is a measure of the effectiveness of f as a predictor of T. The first term on the right hand side (bias) is a measure of how well the estimator has fit the data on the average, while the second term (variance) measures how much change there is in the estimator from sample to sample (“smoothness” of the fit). Many other decompositions are possible (particularly in classifiers, Breiman (1996)), and this kind of breakdown has proved most fruitful, especially in the theory of multiple networks as seen below.

In the machine learning literature bias usually designates those aspects of model structure or the algorithm which enable any inductive learning procedure to generalize beyond its training set. Roughly speaking, we need to introduce bias (structure based on some prior knowledge of the particular domain) in order to compensate for having insufficient samples to populate the input space to the required density. If the bias of a model is incorrect (the class of functions the estimator is drawn from does not include the required regression function) it will not be able to capture sufficient of the statistical regularities in the data. Conversely, a model with greater complexity may overfit the data - hence the variance contribution to the error is high. As has often been remarked, finding the correct bias (typically by using a specific data representation) amounts to solving the problem. In effect, true model-free inference is impossible because of the (infinite in the limit) number of parameters that have to be estimated. In this case the size of the data sets needed to reduce the error due to variance becomes prohibitive.

In this study early stopping via a hold-out set (of at least the size of the training set) has been used, despite its lack of a theoretical basis, for a number of reasons: (1) k-fold cross-validation, particularly for k equal to the data set size, is too computationally intensive for large data sets, as is Bayesian estimation. (2) There is no data shortage in this domain - including unlabelled data gives access to the entire English vocabulary. (3) Error reduction on the training set is not that important for this study, since 100% correct performance can easily be obtained on all 19997 words, at the expense of poor performance outside this set. (4) We are principally interested in the total error reduction that can be achieved (on the unseen data) using the methods described here, and the nature of the remaining error, not in a theoretical optimum level of error on a particular data set.

4.1 Generalization and Error Measure

The error function used in experiments reported on here is the standard sum-of-squares as used in backpropagation for a set of q input-output pairs $D = \{\mathrm{I}_k^q \in \{0,1\}^k,\ \mathrm{T}_i^q \in \{0,1\}^i\}$, where I and T are related to distance metrics on the input and output spaces, as discussed above. This can be formulated in terms of a two-layer network as

$$E = \frac{1}{2}\sum_{qk}[T_k^q - f_s(\sum_j U_{kj}.\ f_s(\sum_i W_{ji}\mathrm{I}_i^q))]^2 \qquad (3)$$

where W_{ji} and U_{kj} are weights from input to hidden and hidden to output respectively, and f_s ( ) is a continuous differentiable function, here the logistic $\frac{1}{1+e^{sum}}$. Bias terms on each unit are omitted here for conciseness.

However, it is better to consider the error apart from the network dynamics because backpropagation is only one of a number of error minimisation techniques, and to focus on external issues such as performance on unseen data. Thus, to make clear the dependence on a particular training set of finite size, it can be written (as derived in Bishop, 1994) relative to a set of n input vectors x and corresponding target vectors t indexed by q in the form

$$E = \frac{1}{2n}\sum_q \| f(x^q) - t^q \|^2 \qquad (4)$$

Essentially there are two methods for controlling the bias and variance of a model relative to a particular data set D: formal and structural stabilisation (Denker et al, 1987). Structural methods typically involve curve-fitting by limiting the number of free parameters in the network, so limiting the degree of the polynomial. Formal methods add an extra term to the error function, which penalises complexity in the model. Bishop (1994) has shown that training with noise is equivalent to regularization of the form:

$$\tilde{E} = E + \lambda\Omega(f) \qquad (5)$$

where $\tilde{E}$ is a modified error function consisting of the original error plus a proportion λ of a penalty term Ω(f) (often based on weight decay). The parameter λ dictates the bias-variance trade-off by controlling the effect of the regularizing function Ω(f) on the minimizing function f(x). Ω(f) is usually chosen on the basis of some prior knowledge concerning the mapping to be learned. For example Bishop (1991,1993) points out that if the mapping is known to be smooth Ω(y) may be chosen to be large for functions with large curvature. It remains a problem to determine the appropriate degree of smoothing from the data alone. Experiments with a modified error function based on combining two networks are described in §5.2, but using knowledge external to the data set.

In fact, there is no reason not to combine a penalty term with structural methods (Williams, 1994). However, as remarked above, Bayesian methods are still nowhere near as computationally efficient as stopped training as a matter of practice. In principle, approaches which make use of noninformative priors are unlikely to be competitive with those which have access to an informative prior. The real issue, therefore, is how to build prior knowledge (whether symbolic, or derived from statistical analysis) of a particular problem into a non-parametric model, preferably in an automated fashion so that it can easily be applied to different data sets.

In the case of the text-to-speech data we have no prior knowledge of the underlying functional form of the distribution of the data which could be directly applied to a surface-fitter such as a neural network, although a great deal of prior knowledge of this task exists at a symbolic level. Attempts to encode some of this knowledge were described in §3. In the absence of information on the underlying function to be fitted, there is no way to determine the optimum number of weights except by experiment (although in this problem this is found to be not a critical issue, as also found by Dietterich and Bakiri (1995). Similarly, smoothing remains a basically unsolved problem, since the model complexity can always be increased in advance of the amount of data available.

4.2 Combining Multiple Networks

The very idea of model selection implies that a best single model must be selected from a competing set. In fact, combinations of models may work better in certain problems and give the user more control than a large single network. There is now a large literature: for approaches which work from the data alone see Breiman, (1992) and references cited; another family of methods is based on "mixtures-of-experts" (e.g. Waterhouse and Robinson, 1994) which uses some prior knowledge or heuristic to determine architecture prior to training.

We can also make use of the combined approach in determining an optimum number of passes through the corrector networks described in §5.1. Krogh and Vedelsby (1995) have shown that the generalization error for multiple networks can be expressed as:

$$E = \overline{E} - \overline{A} \qquad (6)$$

where $\overline{E}$ is a weighted average of error from each network over all patterns, and $\overline{A}$ is a similar weighted sum of ambiguities, where the weights w are positive and sum to one. Equation (6) derives from similar reasoning as the bias-variance trade-off given in (2), except that the averaging is done over network outputs, rather than possible training sets. In the following, as above, for simplicity outputs and targets are assumed to be scalars. The bars are used to denote weighted averages over all networks. The quadratic error ε on a pattern x for a network n is given in the usual way by $\varepsilon^n(x) = [T - f_n(x)]^2$. For the multiple networks:

$$e(x) = [T - \overline{f}(x)]^2 \qquad (7)$$

where $\overline{f}(x) = \sum_n w_n f_n(x)$. The ambiguity, or disagreement among networks, on input x for a single network n is defined as

$$a^n(x) = [f_n(x) - \overline{f}(x)]^2 \qquad (8)$$

The ambiguity for multiple networks is given by

$$\overline{a}(x) = \sum_n w_n a^n(x) = \sum_n w_n [(f_n(x) - \overline{f}(x)]^2 \qquad (9)$$

The fact that the weights w sum to one means that

$$\overline{a}(x) = \sum_n w_n \varepsilon^n(x) - e(x) \qquad (10)$$

Calling the weighted average of the individual errors $\overline{\varepsilon}(x) = \sum_n w_n \varepsilon(x) - e(x)$, we then have

$$e(x) = \overline{\varepsilon}(x) - \overline{a}(x) \qquad (11)$$

Using capital letters to stand for averages over the input distribution we have the ensemble generalization error equal to:

$$E = \sum_n w_n E^n - \sum_n w_n A^n \qquad (12)$$

which can be written in short form as (6) above. From (6) it is clear that $E < \overline{E}$. Where there is no reason to believe more strongly in any of N individual networks (weights equal to 1/N)

$$E \leq \frac{1}{N} \sum_n E^n \qquad (13)$$

where $E^n = \int \varepsilon^n(x) P(x) dx$. Similarly, $A^n = \int a^n(x) P(x) dx$ and $E = \int e(x) P(x) dx$. A key point is that $\overline{A}$ can be estimated using unlabelled data - we do not require the target value T for a given input x. Krogh and Vedelsby use these ideas in conjunction with cross validation, using non-overlapping test sets to determine an optimum N (number of networks) and K (size of test set). They also suggest using the method in an active learning scheme to determine which patterns contain maximal information about the function to be fitted. From (11) it follows that

$$\overline{\varepsilon}(x) \geq \overline{a}(x) \qquad (14)$$

Therefore a pattern with a large ambiguity will always have a large average error. These ideas can also be used with error correcting autoassociative (AA) networks in order to determine the optimum number of cycles (since obviously we cannot make use of the target values to do this). Each network feeds to an identical AA circuit, so that following one cycle each of the outputs will be individually corrected to a new output. Networks with similar outputs are likely to move to similar regions of activation space following the correction, but where there are significant differences these are likely to be exaggerated, unless the inputs are in the close neighbourhood of examples in the AA training set D_{PT} (see §5.2 below). The network disagreement (ambiguity) provides a self-adjusting parameter for controlling the amount of correction to be applied (results are given in §6 below).

It can be seen that the combination of multiple models will be most useful when they disagree most. In the context of a phonetic task a pattern with a high level of ambiguity is likely to mean an irregular spelling, since the relevant contexts will not appear uniformly across each training set -in this study each network is trained on different data. If we assume that the networks have not overfitted their data we may take the training errors themselves as estimates for E^α. This assumption is not likely to be 100% correct, but some of the overfitting can be averaged out in combining networks (see e.g. Perrone and Cooper, 1992, and references therein).

5. System Architecture

In terms of neural networks, the proposed architecture consists of a main network having one hidden layer, and a corrector network (referred to as an AA network because it is auto-associative), with two further SOM stages trained separately (fig. 1). The complete system produces speech from text, but in this report only results from the text-to-phonetic modules are given. In the MLP and corrector networks the arrows represent fully feed-forward connections between every unit in the layers. In figures 3 and 4 shaded arrows represent fully connected layers, whilst single lines represent only connections from each unit to itself.

There are two ways in which the corrector network could be trained, which might be termed (1) joint on-line learning (training alongside the main network at the same time) and (2) separate learning with noise. In the case of (1) the AA units, initially untrained, learn to correct the output from the main MLP. Training time is not reduced by the AA units, but robustness is increased in the operational mode (feed-forward stages only). In the case of (2) each circuit is trained separately, the AA circuit on a specially chosen data set based on frequently occurring words (see §7) , and the main network on data randomly drawn without replacement from the main adapted S&R database (excluding the 4000 word test set). Both procedures give similar results, although method (2) is less sensitive to initial parameters.

<fig 1 about here>

Fig 1. SOMtalk architecture

5.1 Corrector Networks

It has long been hypothesised that learning letter-to-sound rules requires the exploitation of morphemes - sequences of commonly occurring letters (e.g. Klatt, 1987). It is also clear that in any particular language only certain sequences of sounds are phonologically possible. Within this set only a subset will actually be words of the language. With regard to phonemic structure, CV and VC transitions are much more common than CC or VV combinations, especially if one is only concerned with individual words, as in these experiments, and not continuous text. Within this subset, certain CV combinations are more common than others, and so on. Speakers have all this kind of phonological knowledge before they learn to read aloud. The philosophy is that a network should learn to acquire this type of structure, rather than trying to specify this type of prior knowledge in advance. The AA network has been able to learn some of these types of dependencies, so that the 'corrected' output obtained tends to have a CVC structure, even if the transcription is not correct.

Equally, aspects of the task require 'anti-generalization': (1) for example, the letters 'ou' can map onto very different sounds depending on only a very slight change in context: e.g. 'though' `(D@U)` and 'thought' `(TOt)`, and even more difficult cases can occur where the of context may be even more remote from the vowel affected, as in the case of 'photograph' `(f@Ut@grAf)` and 'photographer' (f@t0gr@f@R).Conversely (2) very different inputs may map onto similar outputs, e.g. 'bought', 'caught'). Fig 2. illustrates situation (1), showing an idealised 2D section through activation space.

<fig 2 about here>

Fig 2. Section through two neighbouring attractor basins, showing behaviour near separatrix.

The creation of attractor spaces can address both these issues. It has been found experimentally that, for the output encodings used here, the auto-associative mapping for a given training set can be learnt in about one third to one half of the epochs needed for the hetero-associative mapping (the grapheme-phoneme mapping as described in 2.1 above).

It is important to note that these issues also apply outside the realm of the grapheme-phoneme conversion problem in speech synthesis generally; that is: in any rule-based system which makes use of a segmental phonetic notation (this is all current systems which have moved beyond the prototype stage). This is mainly due to the use of articulatory features to distinguish phonetic segments (inherited from the Chomsky-Halle framework) which are not well-suited to handling variability beyond the context of the articulatory effects of the neighbouring vowel or consonant. Neither, it must be said, is the diphone approach: however the SOMtalk framework can in principle handle these longer-range effects, because a phoneme-based notation is not fundamental to its operation. Any notation based on any phonetic features, articulatory or otherwise, could be used, providing it can be represented in 2 or at most 3D. The number of classes of sounds can be allowed to vary according to the chosen task (or vocabulary) and the number/type of concatenation units used in the output speech.

Attractors have been formulated in terms of energy functions (Hopfield, 1982), in terms of multi-layer sequential networks (Jordan, 1986; Elman 1990) and in the context of backpropagation through time (Werbos, 1990). The simplest way to develop a sequence recognition capability is to unfold time in space (as in the original NET-talk, and in Time Delay Neural Networks (Waibel et al, 1989). However, the latter is only suitable for multi-class classification problems, and does not provide any sequence reproduction capability. A recurrent network is needed to create attractor basins which lead to a sequence reproduction capability. Only a limited reproduction capability is required for two purposes: (1) to capture phonetic regularities which are not easily detectable in the orthographic representation, and (2) to provide extra robustness for frequently occurring combinations - the corrector capability, especially if they require 'anti-generalization' as in fig. 2.

5.2 The Knowledge Transfer Problem

Conventional statistical techniques such as ordinal data analysis, as used by van Santen (1992), are one way to draw out regularities from a speech database, but it is not at all clear how to incorporate these in a neural network (or any learning) scheme. As discussed previously, some bias can be built in using specific representations. The problem is one of transferring knowledge at a symbolic level into a nonparametric model -this might be referred to as the problem of the opacity of neural networks, or more generally the Knowledge Transfer Problem. Within symbolic AI combining knowledge from different sources into the same system has sometimes been referred to as the knowledge folding problem.

A possible solution to KTP is to attempt to learn and encode the prior knowledge in a separate network, here termed a corrector network. This can be done by pre-training a separate circuit on some data which represents a part of the task to be learnt, chosen on the basis of knowledge of the domain. The modularity (and hence re-usability) of the data is here assumed to exist at the level of acquisition, By using this methodology different training set can be chosen appropriate to the particular phonetic task at hand, possibly using different networks for each C_nV and VC_n combination, whose output can then be combined. Experiments in this study have used two types of corrector module: (1) a separate circuit trained with noise (equivalent to formal regularization), see §5.4, and (2) a recurrent network which learns alongside the main network, see §5.3. Both are trained on a data set of m output patterns $D_{PT} = \{ T_i^m \in \{0,1\}^i \}$, and used to modify the output of a network trained on the set D (as above), where $D_{PT} \cap D = \varnothing$. D_{PT} can be chosen to be in some typical of, or basic to the domain in question - in the present case it consists of words which are more common (according to the Kucera-Francis measure) than the rest of the data.

The error measure for a main network MAIN and corrector network PT on n patterns can be written:

$$E = \frac{1}{2n}\sum_q \left\| f_{MAIN}(x^q;D) - t^q \right\|^2 + \alpha \left\| f_{PT}(f_{MAIN}(x^q;D);D_{PT}) - t^q \right\|^2 \qquad (15)$$

The extra error can be used during training as a ‘hinting’ procedure to speed convergence (though this is not a critical issue)by having α start close to unity and decline linearly. More importantly, the function f_{PT} (.) may be taken as giving the final output, so that it operates as a corrector network as described below. In fact, because of the large training sets used an on-line version of backprop is used, so the measure E is not strictly minimised by steepest descent, but by a stochastic approximation thereof.

In a sense, generalization is a non-issue once it becomes clear that all performance is measured relative to a given domain and consists of interpolation within the data space covered by the set of training examples. In the purely non-parametric framework extrapolation to other regions results in a high variance with unpredictable results -unfortunately extrapolation is usually the most interesting form of generalization. Once again, the only solution is to build in the important generalization -in this case by using a specific phonetic representation.

5.3 Training the AA circuit

Shaded arrows indicate fully interconnected units and the single line indicates a connection from each unit to itself. The circled 1s mean that input is taken from the previous time step. The AA units therefore are updated as follows

$$AA_j(t + 1) = \alpha AA_j(t) + OUT_j(t) \quad (16)$$

where $OUT_j(t)$ are the output activations at time t and α is the amount of decay ($\alpha < 1$). The closer α gets to 1 the further back in time the weighted moving average is extended, obviously at the expense of more detailed recall of the immediately preceding time steps. In these experiments the average word consists of 8 phonemes, but in general the immediately preceding 2 or 3 phonemes are most significant in forming attractor basins. Orthographic dependencies may spread back or forward further, but they are taken care of by the 10 letter input window.

There is a recursive cycle in which the encoder network is fed information about its past state (i.e. from its own outputs at one time step previously) so that attractor spaces will develop given enough training. The main objective of this procedure is to increase robustness in performance and to reduce some of the load on the main network.

<fig 3 about here>

Fig 3. Correction performed by encoder network (on-line training)

The procedure amounts to a rudimentary form of knowledge transfer between neural network modules, the latter being one of the most important problems in the neural network field and in Artificial Intelligence generally. This problem has remained unsolved in symbolic AI, but there are number of ways of tackling it from the connectionist viewpoint because it can be treated as a problem in learning.

5.4 Training with noise

Fig. 4 shows how the AA units and main network may function as two separate systems. In the case of separate training noise is added to the input values (but not the targets) on a Gaussian basis over a range of 0 to +/-0.45. The final output is calculated from the forward stages 1 and 2 of the main network, and the result then passed through the AA network.

<fig 4 here>

Fig 4. Encoder network decoupled from main network

One object of using this procedure is to try to make use of prior knowledge, or ‘hints’ from the AA units to guide the main network through the use of a combined error measure (see §5.2). In addition to the theoretical motivation for the use of prior knowledge or structure, their use was prompted by experimental findings that (using the encodings described in §2.2) the AA units, when pre-trained in isolation, could converge on a given training set some 3-4 times faster than the a MLP could learn the grapheme-phoneme mapping for the same data set.

6. How many cycles through the corrector network?

With both the above types of corrector network a problem is to determine the optimum number of correction cycles to produce the best output. Obviously we cannot (except for comparison purposes) make use of the actual target values (“divine guidance”), or the procedure becomes no more than an error accounting scheme, useless for data without target values. Fig 5 shows the number of cycles used in the optimal divine “guidance scheme” where actual error is used to determine the optimum number of cycles, to a maximum of 10 (although in fact never more than 7 are used). The following schemes have been tried:

1. Use a fixed number n of cycles. Values from 1 to 5 have been tried, but this offers the worst performance improvement in terms of overall error, about %1-3.

2. Cycle until the output is stable (no change for e.g. 3 cycles), using a maximum permitted 10 cycles (as with all the methods which vary the number of cycles on a per-pattern basis). This can be done at the level of output activation or output diphone.

Problems are that it is difficult to detect cases where no cycles will help, and the “pathological” cases in which the output cycles into an illegal phoneme location and gets stuck. As a result performance is not good with this procedure (improvements of 5-10 only).

3. Cycle until the distance to the nearest legal phoneme position (for each of the two out put phonemes) is at a minimum. Fig 6 below shows the numbers of cycles used to minimise the phonetic distance. Compared to using the MSE, more cycles are used and performance is not as good.

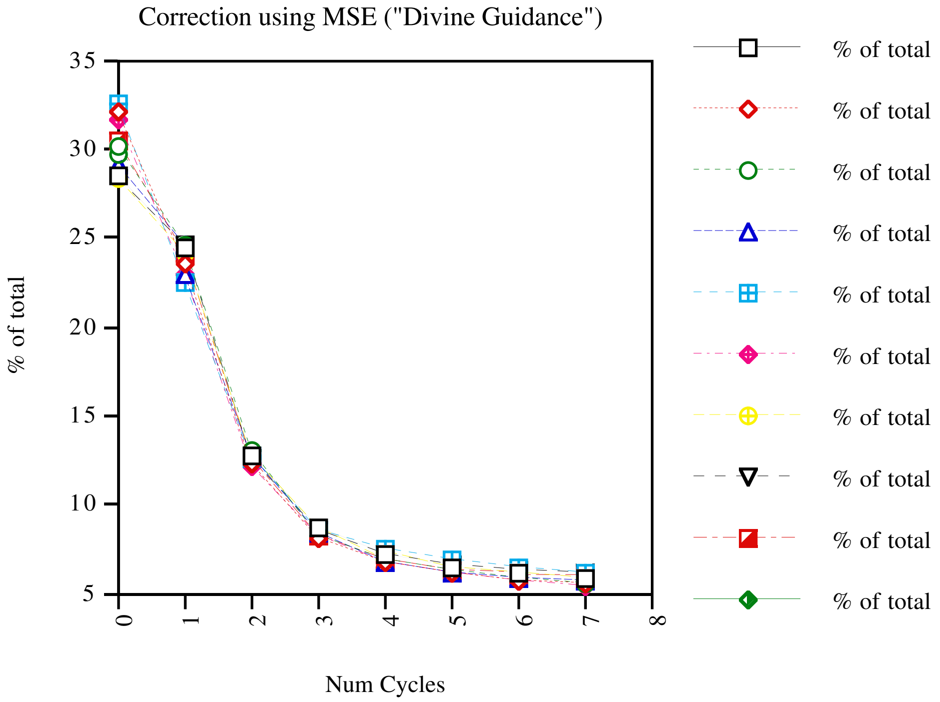

Fig. 5 No. of cycles using MSE for 10 networks

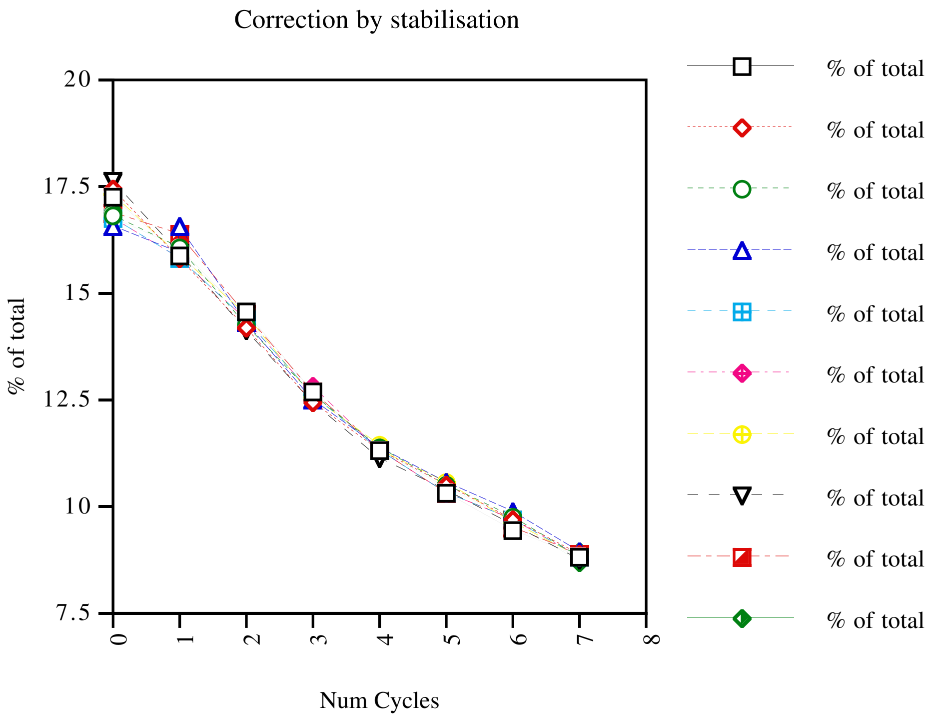

Fig. 6 No. of cycles using phonetic distance for 10 networks

In the case of multiple networks we have additional options:

4. Maximize ambiguity, regardless of error.

This involves re-calculating the ambiguity for all networks at each cycle on a particular pattern (see equation (8) above). Figure 7 shows that in this case the percentages of the 10 networks are much more varied, as would be expected. The individual network errors are higher than before, as we expect patterns.

5. Maximize ambiguity, while trying to hold error low through one of the above stabilisation procedures.

6. Minimize ambiguity instead of maximising.

7. Apply any of procedures 1-3 on the final output, after combining the multiple networks.

In practice it was found that there was little or no advantage to be gained from using the corrector network on the combined output. There is also no guarantee that the stabilisation procedures will actually hold error low, though it prevents the pathological cases referred to above. Used alone they allow too many corrector cycles, which can actually increase overall error on the individual networks, although the error is always reduced when the networks are combined. We wish to produce the maximum disagreement between the networks, but not allow illegal phoneme strings (which could have a high ambiguity since they are not in the neighbourhood of corrector network examples) to distort the final output.

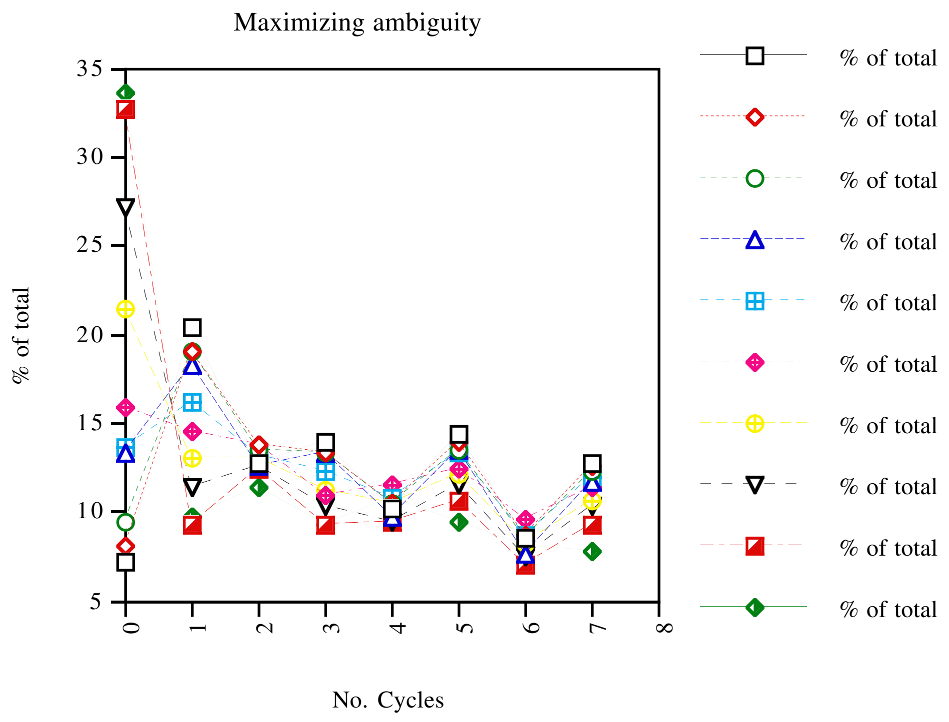

Fig. 7 No. of cycles used to maximize ambiguity

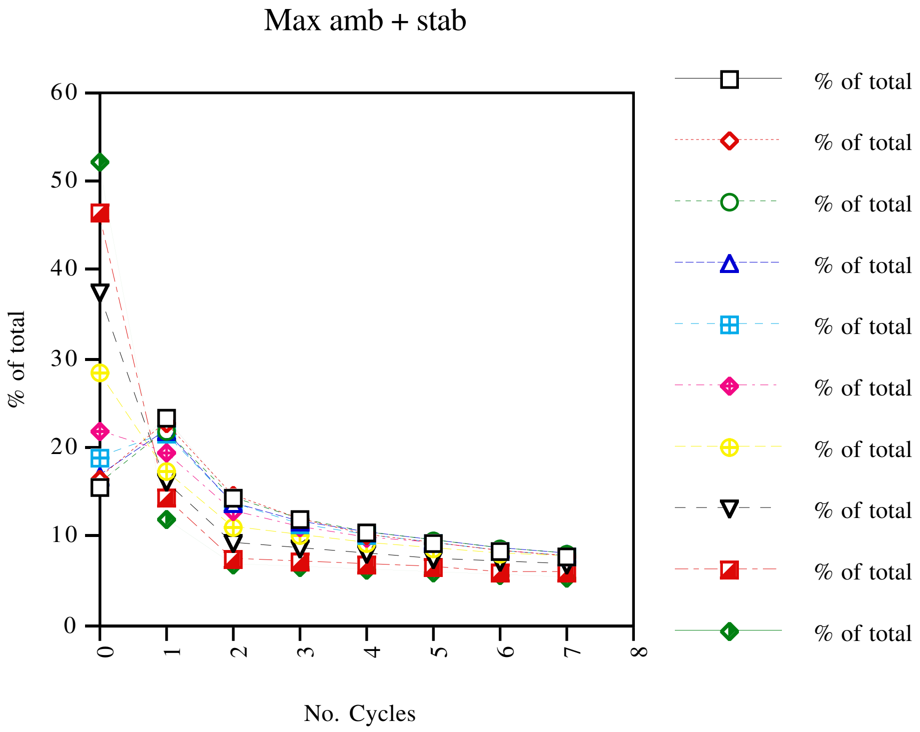

Fig. 8 No of cycles to maximize ambiguity and stabilize

Results based on minimising ambiguity in conjunction with the same set of procedures as above (see Fig. 9) produces similar results

7. Training procedure and results

Initially the methodology has been applied to the grapheme-diphone conversion problem to measure the viability of embedding prior knowledge in the way described in §3. Two sets of 4000 words were drawn at random (without replacement) from the original NETtalk training set, which made up the training set and test set for each run on each architecture. In order not to optimise for a particular test set, further (non-intersecting) test set were used for network configuration tried. Apart from this, the standard cross-validation procedure was used to prevent over-fitting the training set. Only the number of hidden units were varied, as using two hidden layers was not found to bring any increase in performance. A third set of the 4000 most frequently used words (according to the Kucera-Francis measure supplied with the MRC dictionaries) were used to train the AA circuit. None of these words appear in the test sets.

The important fact is not so much the total error (as no great effort has been made to optimise for a particular test set) but the amount of correction offered by the AA circuit and other correction schemes. The training set error was typically about %5-10 better at the time of peak performance when training was stopped. There was no problem achieving %100 correct on the training set by continuing to train past this point.

§7.1 Results on the individual networks

The procedure for determining the output was to cycle through the AA circuit until no changes in output phoneme have occurred for 3 iterations. All the results given in table 1 are for the test set (data unseen during training). A typical test set of 4000 words gave 29551 pattern pairs (10 letter input window to two phoneme output window) which equalled 59102 individual phonemes to learn. Hence, e.g. a 5 percent improvement amounts to an extra 29551.1 phonemes correct in this context.

In tables 1 and 2 bit correction refers to the procedure 2(a), as described in §3.1. The first column gives the number of hidden units. These results represent typical runs of about 15 -25 presentations of each pattern, stopping when a five-point moving average of error on a cross-validation set (of size equal to the training set) begins to rise. No substantial effort has been made to get the uncorrected error as low as it might be, given these architectures. The error represents the percentage incorrect on the unseen test data.

No. Hidden	No Correction	Bit correction	AA circuit active	Total Error Reduction
65	29.49437	28.80937	25.09502	4.39868
70	42.57896	41.69796	36.89604	5.68292
80	38.12932	37.76832	32.83191	5.29741
120	34.18124	34.04024	29.37183	4.80941
140	29.40287	28.87187	25.96259	3.44028
160	31.03633	30.46933	25.66436	5.37197
180	36.66870	36.52770	30.89642	5.77228
140-140	45.71702	44.74102	41.92780	3.78922

Table 1: Some results for single networks: Percentage of test set wrong: Typical results using structured SOM representation to obtain target co-ordinates (see appendix 1).

No. Hidden	No Correction	Bit correction	AA circuit active	Total Error Reduction
120	68.252677	68.21332	68.005287	0.24739

Table 2: Percentage of test wrong: Results on randomly generated co-ordinates to obtain target values.

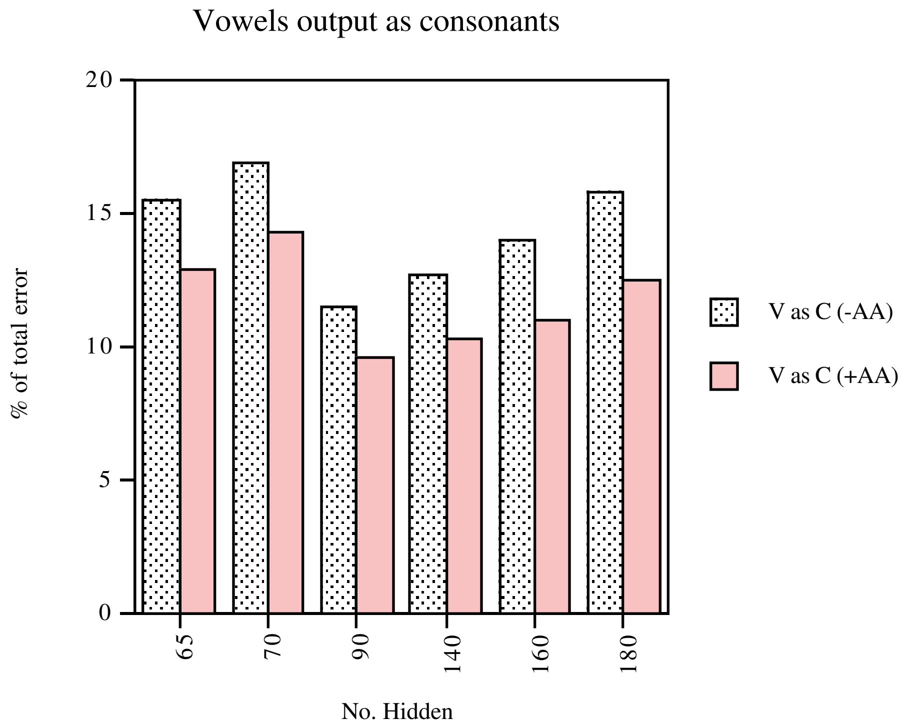

Fig 4: Effect of AA units on composition of error for different size networks: vowels misidentified as consonants. +/- AA shows AA units used/not used.

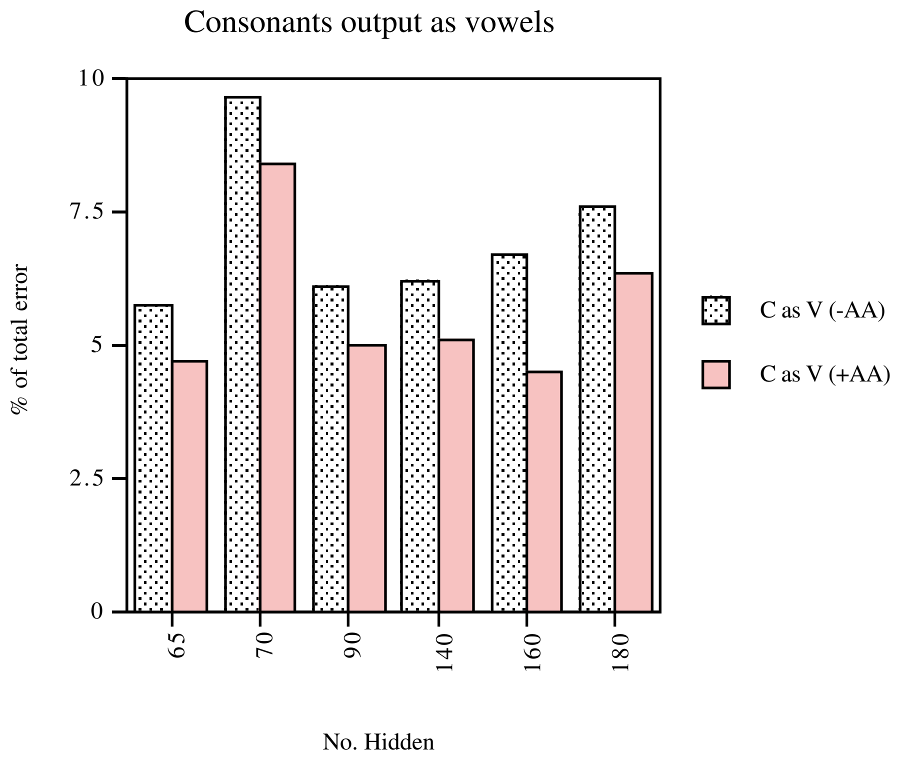

Fig 4a: Effect of AA units on composition of error for different size networks: consonants misidentified as vowels.

No. Hidden	+/- AA	% V as V	% V as C	% C as C	% C as V	Total error
120	-AA	3.073743	5.5747759	46.539921	13.054087	68.242511
120	+AA	3.175410	5.720483	46.268808	12.833808	67.998509

Table 4: Effect of using random target co-ordinates on composition of error (values in percentages of total error)

Table 4 shows the importance of the D_{phon} representation, as base error is far higher and the reduction achieved with the AA circuit much smaller.

The basic principle is shown to work, in that correction of the output takes place, but the effect is quite small at this stage. Further work is needed to refine the exact order of each stage in calculating the output, and the influence of each stage upon the next. Some of the issues are discussed in §7.2, where the predicted error of various combinations is shown. Breakdown of the error shows the effect more clearly in that it can be seen that the effect of learning something about the CV and VC structure of the data makes a vowel less likely to appear as a consonant and vice versa. In the other categories (vowels misidentified as a different vowel, and consonants misidentified as a different consonant) the error reduction is smaller.

§7.2 Results with multiple networks

Each of the N networks is trained on a non-overlapping partition D^n of the training set D, such that the final output will be given by:

$$f(I;D) = \sum_{n=1}^{N} w_\alpha f_n(I;D^{(n)}) \qquad (18)$$

where

$$\sum_n w_\alpha = 1 \qquad (19)$$

If the p.d.f of D is known, we may simply minimize the MSE to obtain the optimal combination of coefficients w_α. Alternatively Krogh and Vedelsby (1995) suggest a linear programming procedure such that each network contributes exactly $w_\alpha E$ to the generalization error of (6). If estimates of E are used unlabelled examples can be used to learn w_α by gradient descent. If the input distribution is not known (the usual case), a possibility is to resort to least-squares fitting such that

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left\{ \sum_k \left[t_k - \sum_n w_n f_n(i_k; D^{(n)}) \right]^2 \right\} \quad (20)$$

However, we do not wish to rely on knowing the target values t, so a simpler method is used in scheme b below to give the highest weights to the networks which disagree most with the others, where each weight is simply set to

$$w_n = \frac{a^n}{\sum_n a^n} \quad (21)$$

at each iteration through the patterns, where the ambiguity a is altered each time as the weighted output changes. This ensures the largest weights go to networks with the largest ambiguities, whilst conforming with (19). To get the opposite effect:

$$w_n = 1 - \frac{a^n}{\sum_n a^n} \quad (22)$$

The original 20,000 word DECtalk training set was split into 11 subsets, giving 1818 words for each of the 10 networks, and leaving 1818 as a test set. The AA circuit is trained as before (hence there is some overlap with each individual training set). The multiple networks all had the same number of hidden units: 120. The legends are as follows (refer to §6 for full explanations):

a. No adjust: standard networks, no correction used.
b. Adjust we: the weights w_α are adjusted according to (21)
c. Max. Amb: Ambiguity is maximised with respect to the other networks
d. Stabilize: cycle until phonetic distance minimised.
e. DG: Divine guidance, the best performance possible.

Unless otherwise stated the weights w_α are uniform. If the w_α are not uniform this affects results in the case of the single networks because the ambiguity is recalculated based on the modified output (produced from the weighted average) on a per-pattern basis. A good correlation between ambiguity and error is seen in figs 10 and 11. Fig 12 shows the value of E-A for each network. These values can be used as estimates for the multiple generalization error

Low ambiguity indicates a high bias in the overall combined networks, since they even agree on patterns outside their training set. In this case we may expect the final combined generalization error to be very similar to the weighted average of the individual errors. High ambiguity indicates a high variance, and in this case we can expect the combined generalization error to be less than the weighted average of the individual network errors.

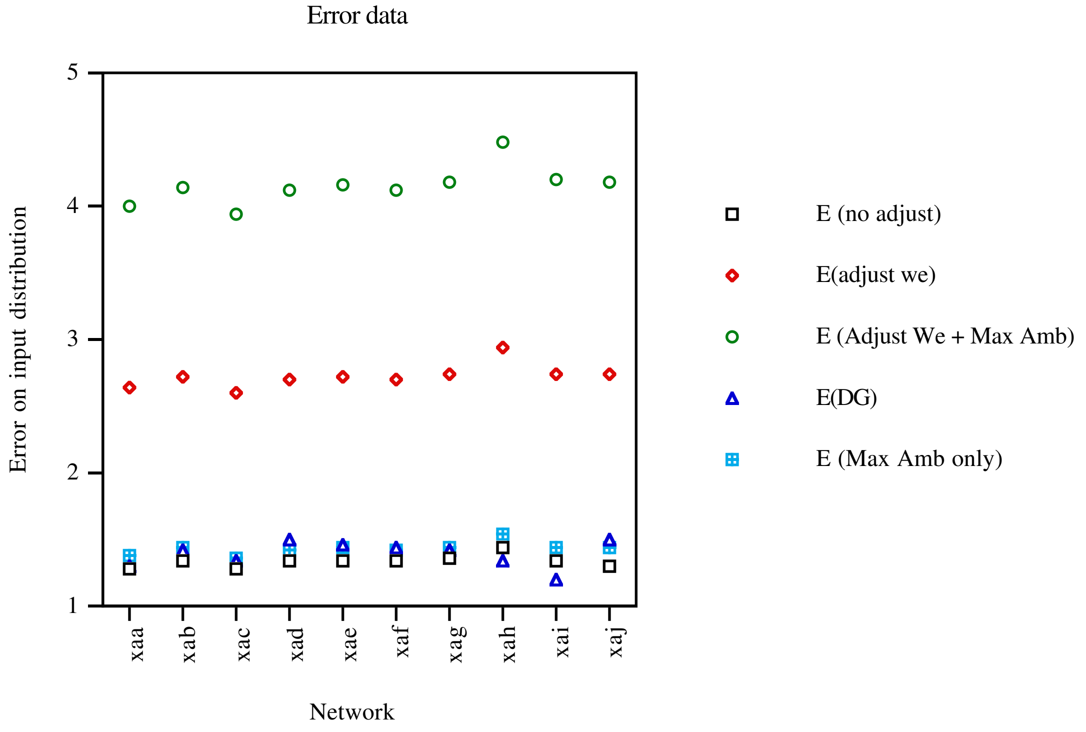

Fig. 10: MSE averaged over the input distribution for the 10 networks, using various correction methods.

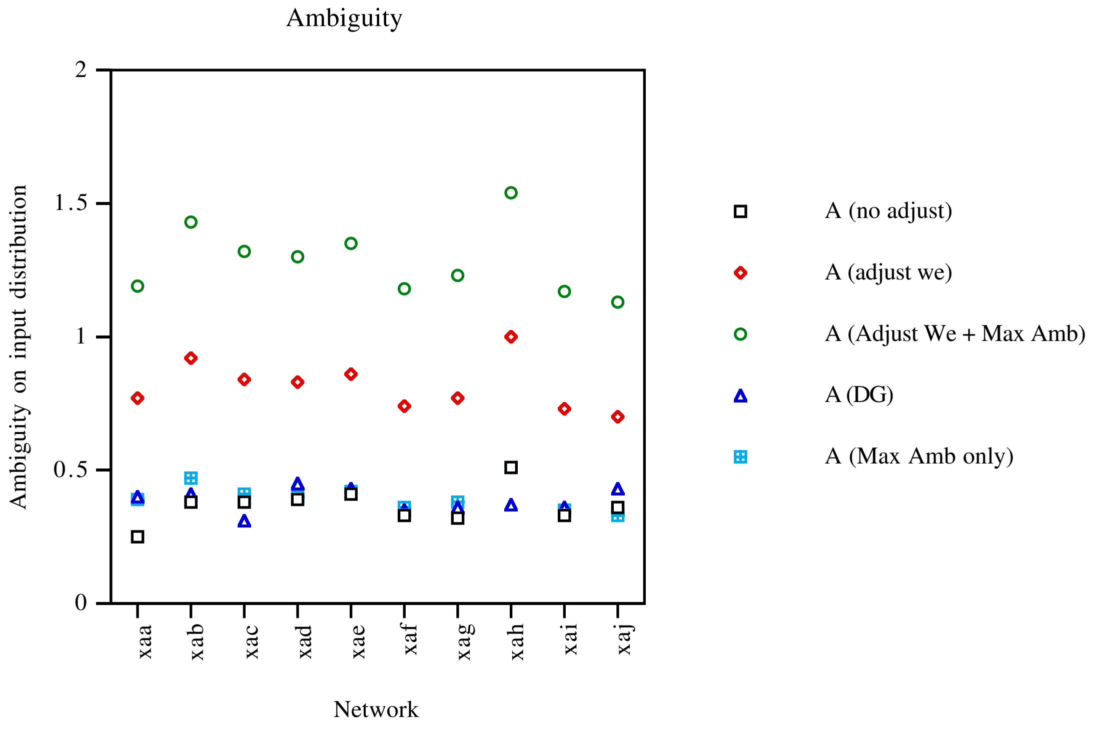

Fig. 11: Ambiguity on a per-pattern basis on the test set for 10 networks, using various correction methods

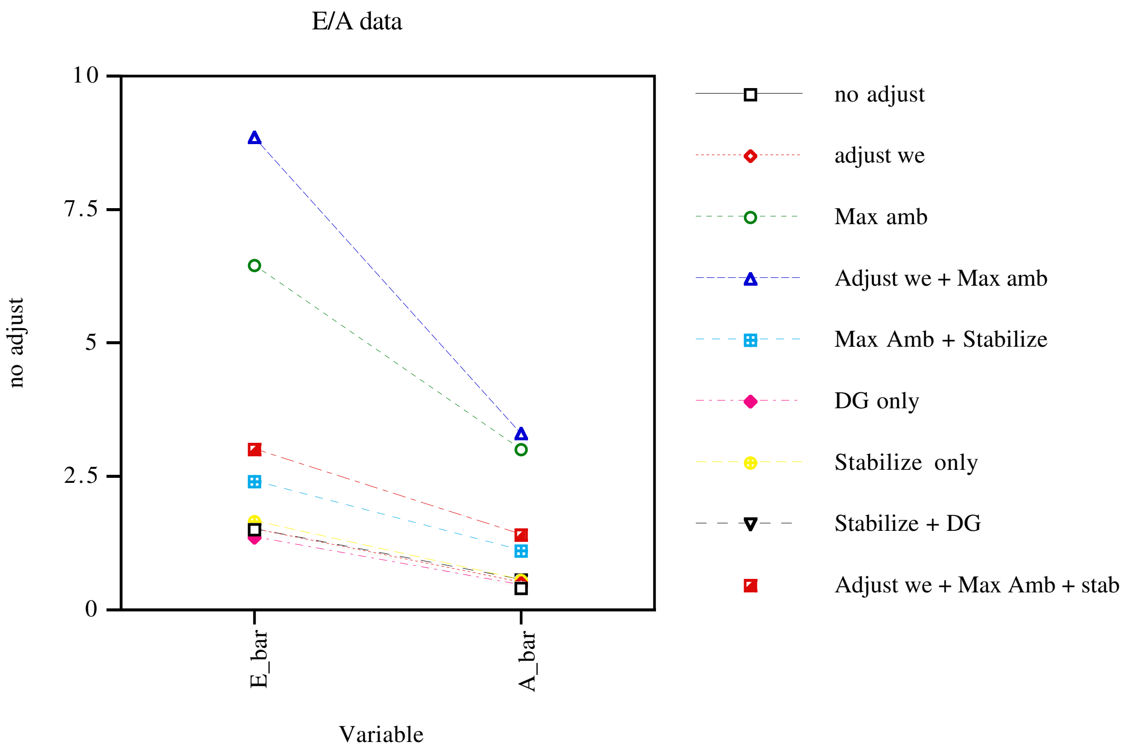

Fig 12: Results for all methods on the combined output.

Overall actual error levels are in line with the predicted error levels of fig 12. The flatter the slopes and the lower the error the better the predicted performance ($\overline{E} - \overline{A}$ reduces). MSE does not always correspond exactly with error at the phoneme/diphone level, but it is not possible for them to drift too far apart. The best performing methods are shown in fig 13 below. These correspond to percentage error rates of %15-20, but it is believed these could be further reduced by more careful training of the individual networks, e.g. using leave-one-out cross-validation, though this is not our concern here.

A reduction in error compared with a single network is seen with the above methods, though not to the extent seen in the stacking procedures for combining multiple models (Breiman, 1992; Wolpert 1992). These procedures make more efficient use of the data (assuming various issues such as size of cross-validation set are got right), at a greater computational cost, but naturally they do not offer any model or hypothesis of an underlying process. They must be retrained from scratch if new data is to be added, whereas adding extra networks is straightforward in the SOMtalk architecture. We have been able to investigate the use of a set of frequently occurring words and the effect of various interpretable correction procedures. In addition by using the formulae of Krogh and Vedelsby we have an explicit way of balancing individual network error against the network disagreements (ambiguity) which allows us to explore the error correcting properties of a particular representation.

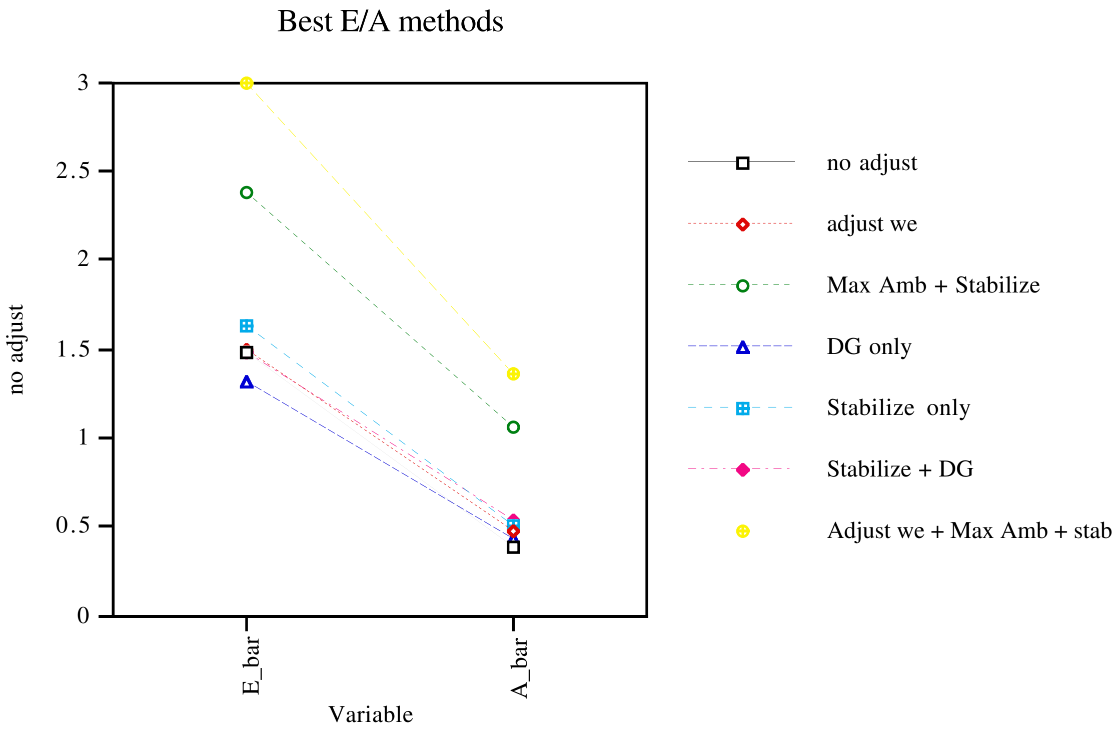

Fig 13: Best methods for combined networks

§8. Acknowledgements

None yet!

9. References

van den Bosch, A, and Daelemans, W, (1993), “Data-Oriented Methods for Grapheme-to-Phoneme Conversion”, in *Proceedings of the 6th Conference of the European Chapter of the ACL,* Utrecht, April 1993.

Breiman, L, (1992), “Stacked Regressions”, Technical Report, Department of Statistics, University of California, Berkeley

Breiman, L, (1996), “Bias, Variance and Arcing Classifiers”, Technical Report 460, April 1996, Statistics Department, University of California, Berkeley, CA 94720

Cohen, A , (1995), “Developing a Non-Symbolic Phonetic Notation for Speech Synthesis”, *Computational Linguistics*, **21** (4), pp. 567-575

Daelemans, W , and van den Bosch, A, (1993), “TABTALK: Reusability in Data-Oriented Grapheme-to-Phoneme Conversion”, in *Proceedings of Eurospeech 1993,* Berlin

Dietterich, T G, and Bakiri, G, (1991) ,”Error-correcting output codes: A general method for improving multiclass inductive learning programs”, in *Proceedings of the Ninth National Conference on Artificial Intelligence*, pp. 572-577, AAAI Press/MIT Press.

Dietterich, T G, and Bakiri, G, (1995), “Solving multiclass learning problems via error-correcting output codes”, *Journal of Artificial Intelligence Research,* **2**, pp. 263-286.

Denker, J, Schwartz, D, Wittner, B, Solla, S, Howard R, Jackel, L, and Hopfield, J, (1987), “Large Automatic Learning, Rule Extraction, and Generalization”, *Complex Systems*, **1**, pp.877-922

Elman, J L, (1990), “Finding Structure in Time”, *Cognitive Science*, **14**, pp.179-211

Geman, S, Bienenstock, E, and Doursat, R, (1992), “Neural Networks and the Bias/Variance Dilemma”, *Neural Computation*, **4**, pp. 1-58

Hopfield, J J, (1982), “Neural Networks and Physical Systems with Emergent Collective Computational Abilities”, *Proceedings of the National Academy of Sciences*, *USA* **84**, pp.8429-8433

Jervis, T T, and Fitzgerald, W J, (1993), “Optimization Schemes for Neural Networks”, Technical Report CUED/F-INFENG/TR144, Cambridge University Engineering Department, August 1993.

Jordan, M I, (1986), “Attractor Dynamics and Parallelism in a Connectionist Sequential Machine”, *Proceedings of the Eighth Annual Conference of the Cognitive Science Society* (Amherst 1986), Hillsdale: Erlbaum, pp.531-546

Kohonen, T, "Self-Organisation and Associative Memory", Springer Verlag, 3rd Ed. 1989

Kohonen, T, "The Self-Organising Map", *IEEE Proceedings* , **78**, 9, pp.1464-1480, 1990

Perrone, M P, and Cooper, L N, (1992), “When Networks Disagree: Ensemble Methods for Hybrid Neural Networks”, in Mammone, R J, ed., Neural Networks for Speech and Image Processing, Chapman-Hall 1993

van Santen, J P H, (1992), “Deriving text-to-speech durations from natural speech”, in Bailly, G, Benoit, C and Sawallis, T R, (Eds) “Talking Machines: Theories, Models and Designs”, Elsevier Science Publishers B.V., 1992

Sejnowski, T J, and Rosenberg, C R, (1987), “Parallel networks that learn to pronounce English text”, *Complex Systems,* **1**, pp. 145-168

Waibel, A, Hanazawa, T, Hinton, G, Shikano, K, and Lang, K, (1989), “Phoneme Recognition Using Time-Delay Neural Networks”, *IEEE Transactions on Acoustics Speech and Signal Processing* **37**, pp. 328-339

Waterhouse, S R, and Robinson, A J, (1994), “Non-linear Prediction of Acoustic Vectors Using Hierarchical Mixtures of Experts”, *Neural Information Processing Systems*, **7**, Morgan Kaufmann 1994

Werbos, P J, (1990) “Backpropagation Through Time: What It Does and How to Do It”, Proceedings of the IEEE, **78**, No. 10, October 1990

Williams, P M, (1994), “Bayesian Regularisation and Pruning using a Laplace Prior”, Cognitive Science Research Paper CSRP-312, February 1994, COGS, University of Sussex

Wolpert, D H, (1992), “Stacked Generalization”, *Neural Networks*, **5**, No. 2, pp 241-259

Wolpert, D H, and Macready, W G, (1995), “No free lunch theorems for search”, Technical report SFI-TR-95-02-010, The Santa Fe Institute